REMARKS

[0001] Applicant respectfully requests reconsideration and allowance of all of the claims of the application. Claims 1-5, and 18-24 are presently pending. No claims are amended herein.

Formal Request for an Interview

[0002] If the Examiner's reply to this communication is anything other than allowance of all pending claims, then I formally request an interview with the Examiner. I encourage the Examiner to call me—the undersigned representative for the Applicant—so that we can talk about this matter so as to resolve any outstanding issues quickly and efficiently over the phone.

[0003] Please contact me to schedule a date and time for a telephone interview that is most convenient for both of us. While email works great for me, I welcome your call as well. My contact information may be found on the last page of this response.

Claim Rejections under § 103

[0004] The Examiner rejects claims 1-5 and 18-24 under § 103. For the reasons set forth below, the Examiner has not shown that the cited reference anticipates the rejected claims.

[0005] Accordingly, Applicant's attorney respectfully requests that the § 103 rejections be withdrawn and the case be passed along to issuance.

[0006] The Examiner's rejections are based upon the following references alone:

- **US Patent No. 7,346,780 to Sinha et al:** *"Sinha et al"* hereinafter, (issued October 9, 2003); and
- **US Patent Publication No. 2005/0132375 to Douceur et al:** *"Douceur et al"* hereinafter, (published June 16, 2005).

Obviousness Rejections

Lack of *Prima Facie* Case of Obviousness (MPEP § 2142)

[0007] Applicant disagrees with the Examiner's obviousness rejections. Arguments presented herein point to various aspects of the record to demonstrate that all of the criteria set forth for making a *prima facie* case have not been met. To establish *prima facie* obviousness of a claimed invention, all of the claim recitations must be taught or suggested by the prior art[1] and "all words in a claim must be considered in judging the patentability of that claim against the prior art."[2] Further, if prior art, in any material respect teaches away from the claimed invention, the art cannot be used to support an obviousness rejection.[3] Moreover, if a modification would render a reference unsatisfactory for its intended purpose, the suggested modification / combination is impermissible.[4]

Based upon *Sinha* and *Douceur et al*

[0008] The Examiner rejects claims 1-5 and 18-24 under 35 U.S.C. § 103(a) as being unpatentable over *Sinha* and *Douceur et al.* Applicant's attorney respectfully traverses the rejection of these claims and asks the Examiner to withdraw the rejection of these claims.

---

[1] *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974)

[2] *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)

[3] *In re Geisler*, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed Cir. 1997)

[4] See MPEP § 2143.01

Independent Claim 1

[0009] Applicant's attorney submits that the combination of *Sinha* and *Douceur et al* does not teach or suggest all of the elements as recited in this claim. In specific, claim 1 recites calculating, on each task change between a first program module switching from foreground to background and a second program module switching from background to foreground, a signature of at least part of the second program module instruction lines.

[0010] For example, according to an embodiment, a CPU may have the capability of multitasking between executing two or more separate and distinct program modules. As one program module is executed and engaged by the CPU, it may be referred to as running in the foreground. Other program modules that are executing, but are not currently engaged by the CPU in the execution of specific tasks may be referred to as running in the background. Thus, in an effort to ensure that a background program module has not been tampered with by malicious software of hackers, each time a program module transitions to become a foreground application in which the CPU is executing instructions from said application, at least part of the application that is switching from the background to the foreground is used to generate a signature. This signature may be compared to a stored signature that was generated and stored in a memory when the application transitioned to the background. If the signatures match, then this match may be interpreted to mean that the program module now transitioning back to the foreground has not been changed or modified. See generally, paragraph [8] of the detailed description section of the specification.

[0011] However, *Sinha et al* teaches a method that has several differences that ultimately teach away from the recitations of claim 1. *Sinha et al* does not teach the recitations of claim 1. In specific, *Sinha et al* is directed, generally, toward a digital rights management (DRM) solution that may verify that two instances of the same program module are unaltered duplicates of each other, (*e.g.*, typically when operating on separate computer systems) using a so-called "integrity veracitication" technique as generally described in the cited and applied section, column 7, line 31 to column 8, line 25 of *Sinha et al*. This method of *Sinha et al* may also check multiple serial execution instances of a single program module against each other over the course of time on a single computer. Thus, an output tracing method may check an execution-identifying signature each time the one program module is executed to verify that the signature matches other signatures stored at previous execution times. Of course, the program must be terminated between each instance of verification (*e.g.*, not simply operating in the background).

[0012] The Examiner correctly acknowledges that *Sinha et al* offers no teaching or discussion about performing such a signature calculation in response to a CPU switching tasks between two different and simultaneously executing programs (*i.e.*, switching an application from background to foreground and vice versa). Moreover, there is no teaching anywhere in *Sinha et al* with regard to foreground and background execution in any capacity and the whole of *Sinha et al* is silent with respect to even the concept of background and foreground execution.

[0013] *Douceur et al* does not remedy this deficient teaching. *Douceur et al* is directed to a system and method for limiting the effects of background tasks on executing foreground tasks as the CPU is a limited resource. Thus, the system and method ensures that any background tasks that are executing at a "groveler" (*e.g.*, a task engine dedicated solely to background tasks) does not interfere with the more important foreground tasks currently executing. Further, the cited paragraphs [0005]-[0006]. [0036], and [00364]-[0065] describe a feature that solely part of the background task engine wherein files may be checked against memory to see if anything has changed since the background task engine was operating on a particular file. This is because the background task engine may be suspended for large amounts of time if the foreground tasks heavily utilize the CPU.

[0014] However, claim 1 recites calculating, on each task change between a first program module switching from foreground to background and a second program module switching from background to foreground, a signature of at least part of the second program module instruction lines. That is, the system of method of an embodiment of claim 1 actually has programs switching from background execution to foreground execution and vice versa. There is simply no teaching anywhere in *Douceur et al* that teaches or even suggests the possibility of tasks in the "groveler" being executed by a different task engine – namely the foreground task engine. Thus, no prior art of record, whether considered individually or in any permissible combination with each other, teaches program modules switching between background execution and foreground execution.

[0015] Inexplicably, the Examiner has argued that a motivation to combine these teachings of *Sinha et al* and *Douceur et al* is to attain a resulting "system for limiting the interference of background process on the foreground process." This logic may, in fact be true, but such a motivation to combine these references in this manner has absolutely nothing to do with that which is recited in claim 1. Specifically, checking signatures of tasks switching from background execution to foreground execution makes irrelevant any impact simultaneously executing tasks may have on each other. Such a twist in logic in a clear indication of hindsight reasoning wherein the Examiner is finding motivation to solve problems that do not exist. As a matter of law, obviousness may not be established using hindsight obtained in view of the teachings or suggestions of the applicants.[1] To guard against the use of such impermissible hindsight, obviousness needs to be determined by ascertaining whether the applicable prior art contains any suggestion or motivation for making the modifications in the design of the prior art article in order to produce the claimed design and not a made-up problem. The mere possibility that a prior art teaching could be modified or combined such that its use would lead to the particular limitations recited in a claim does not make the recited limitation obvious, unless the prior art suggests the desirability of such a modification.[2]

---

[1] *W.L. Gore & Assocs., Inc. v. Garlock, Inc.*, 721 F.2d 1540, 1551, 1553, 220 USPQ 303, 311, 312-13 (Fed. Cir. 1983), cert. denied, 469 U.S. 851 (1984).

[2] *See In re Gordon*, 733 F.2d 900, 902, 221 USPQ 1125, 1127 (Fed. Cir. 1984).

[0016] Further, as mentioned above, *Sinha et al* actually teaches away from the recitations of claim 1 because *Sinha et al* is only concerned with a single serial application execution wherein any calculation is only performed when the program module is first executed. Therefore, there is no possible way to check the program module once it has been executed without first shutting it down again and restarting. Such a limitation is hardly comforting when malicious software is designed to attack currently executing software as is the goal of the recitations of claim 1. Any conclusion that these two references could be combined to accomplish a security task that neither one is concerned with is broad and conclusory. Such broad, conclusory statements do not come close to adequately addressing the issue of motivation to combine, are not evidence of obviousness, and therefore are improper as a matter of law. *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

[0017] Consequently, no permissible combination of *Sinha et al* and *Douceur et al* teaches or suggests all of the elements and features of this claim. Accordingly, Applicant's attorney asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 2-3

[0018] These claims ultimately depend upon independent claim 1. As discussed above, claim 1 is allowable. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Additionally, some or all of these claims may also be allowable for additional independent reasons.

Independent Claim 4

[0019] Applicant's attorney submits that *Sinha et al* and *Douceur et al* do not teach or even suggest all of the elements as recited in this claim. Claim 4 recites a processor of multitask execution of several programs, each of the several programs being different from each other exploiting a table of correspondence, each correspondence being associated with an identifier of the involved program, comprising means for calculating a current signature, and means for comparing this signature with the identifier of the program stored in the correspondence table.

[0020] As discussed above, *Sinha et al* is directed, generally, toward a digital rights management (DRM) solution that may verify that two instances of the same program module are unaltered duplicates of each other, using a so-called "integrity veracitication" technique as generally described in the cited and applied section, column 7, line 31 to column 8, line 25 of *Sinha et al*. As such, teaching that two identical program remain identical to each other when executed on different computer platforms is not the same as checking the signatures of two different program modules executing at the same CPU. Identical programs can never be different. The Examiner correctly acknowledges this and then turns to *Douceur et al* as somehow bridging this gap.

[0021] *Douceur et al* is directed to a system and method for limiting the effects of background tasks on executing foreground tasks as the CPU is a limited resource. Thus, the system and method ensures that any background tasks that are executing at a "groveler" (*e.g.*, a task engine dedicated solely to

background tasks) does not interfere with the more important foreground tasks currently executing. While these two different program modules are executing exclusively in either the foreground or the background in *Douceur et al*, there is no teaching in this reference in this context wherein a signature calculated may be compared to a table of signatures.

[0022] Furthermore, there is simply no disclosure anywhere in *Sinha et al* or in *Douceur et al* that can be construed to teach exploiting a table of correspondence, each correspondence being associated with an identifier of the involved program. Thus, no permissible combination of *Sinha et al* and *Douceur et al* teaches or suggests all of the elements and features of this claim. Accordingly, Applicant's attorney asks the Examiner to withdraw the rejection of this claim.

Dependent Claim 5

[0023] This claim ultimately depends upon independent claim 4. As discussed above, claim 4 is allowable. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Additionally, this claim may also be allowable for additional independent reasons.

Independent Claim 18

[0024] Applicant's attorney submits that *Sinha et al* and Douceur et al, whether considered individually or in any permissible combination, does not teach or suggest all of the elements as recited in this claim. In specific, claim 18,

as amended, recites executing, at a CPU, a plurality of programs simultaneously, each program having a unique signature calculated when first executed and calculating, on each task change, a new signature, and checking the conformity of the new signature with the unique signature. The Examiner acknowledges that *Sinha et al* does not teach calculating, on each task change, a new signature, and checking the conformity of the new signature with the unique signature.

[0025] *Douceur et al* does not remedy this deficient teaching. *Douceur et al* is directed to a system and method for limiting the effects of background tasks on executing foreground tasks as the CPU is a limited resource. Thus, the system and method ensures that any background tasks that are executing at a "groveler" (*e.g.*, a task engine dedicated solely to background tasks) does not interfere with the more important foreground tasks currently executing. Further, the cited paragraphs [0035]-[0036] describe a feature that solely part of the background task engine wherein files may be checked against memory to see if anything has changed since the background task engine was operating on a particular file. This is because the background task engine may be suspended for large amounts of time if the foreground tasks heavily utilize the CPU. Thus, *Douceur et al* does not teach tasks that change.

[0026] Further, *Sinha et al* actually teaches away from the recitations of claim 1 because *Sinha et al* is only concerned with a single serial application execution wherein any calculation is only performed when the program module is first executed. Therefore, there is no possible way to check the program module once it has been executed without first shutting it down again and restarting.

Such a limitation is hardly comforting when malicious software is designed to attack currently executing software as is the goal of the recitations of claim 1. Any conclusion that these two references could be combined to accomplish a security task that neither one is concerned with is broad and conclusory. Such broad, conclusory statements do not come close to adequately addressing the issue of motivation to combine, are not evidence of obviousness, and therefore are improper as a matter of law. *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

[0027] Further, as discussed above, the Examiner is clearly using hindsight reasoning which is impermissible as a matter of law.

[0028] Consequently, no permissible combination of *Sinha et al* and *Douceur et al* teaches or suggests all of the elements and features of this claim. Accordingly, Applicant's attorney asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 19-24

[0029] These claims ultimately depend upon independent claim 18. As discussed above, claim 18 is allowable. It is axiomatic that any dependent claim which depends from an allowable base claim is also allowable. Additionally, some or all of these claims may also be allowable for additional independent reasons.

Conclusion

[0030] All pending claims are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the application. If any issues remain that prevent issuance of this application, the **Examiner is urged to contact me before issuing a subsequent Action**. Please call or email me at your convenience.

[0031] Any additional fees required as a result of this amendment have been paid from the below-referenced deposit account as filed herewith. Should further payment be required to cover such fees you are hereby authorized to charge such payment to Deposit Account No. 07-1897.

Respectfully Submitted,

Graybeal, Jackson LLP
Representatives for Applicant

/Kevin D. Jablonski/ Dated: December 8, 2010

Kevin D. Jablonski (kevin@graybeal.com)
Registration No. 50,401
USPTO Customer No.: 00996

Telephone: (425) 455-5575
Facsimile: (425) 455-1046